Remarks

Claims 1 through 21 are in prosecution. Claims 1 to 6 are rejected as non-statutory matter under 35 USC 101. Claims 1 to 6 and 15 to 20 are rejected as anticipated under 35 USC 102(b) by Cascio et al (US Pub 2002/00918180, hereinafter "Cascio". Claims 8 to 14 are rejected under 35 USC 103 as obvious over Cascio in view of DaCosta et al (US Pub. 2002/0120725), hereinafter "DaCosta".

The Section 101 Rejection

Examiner notes that the language of claims 1-7, directed to a computer storage medium, is sufficiently imprecise so that the claimed steps suggest expending energy, which a passive medium cannot do. Claim 1 is amended to correct this deficiency. Specifically, the preamble of claim 1 now makes clear that the medium comprise computer code contained in the medium for executing certain steps. This type of claim construction has been approved many times in the past for computer storage medium claims. It is requested that this rejection be removed.

The Section 102 Rejection

Cascio and the present application solve the same problem, that of transforming legacy character-based terminal screens into a form suitable for graphical user interface (GUI) applications, without requiring end-user programming. The methods used by Cascio and the present invention are fundamentally different and it is argued that the present claims sufficiently reflect the fundamental difference of this application over Cascio. Cascio describes an algorithm in which the designers specify a set of rules, using regular expressions in one embodiment, that allow a computer process to recognize a portion of a screen by matching a rule to the screen data. At the end of the process, all the pre-specified output formats associated with the matching rules are used to build a transformed GUI screen.

Thus, Cascio uses a single process to match rules to the screen data. Cascio specifies in paragraph 47 that more that one processor may simultaneously process rules against the incoming character stream. But, each computer is still executing the same single process and a set of rules to identify a portion of a screen that matches a rule. The use of multiple computers just speeds-up the process.

In contrast, the claimed invention uses a plurality of processes, called agents, in a computer to recognize screen patterns. Unlike Cascio, each agent is conditioned to match a single screen pattern. In some sense, an agent might be considered analogous to a Cascio rule, but the implementations are is quite different, A Cascio rule is a grouping of words that must be parsed and interpreted by the single Cascio process. But each of the agents in this application merely looks for data patterns and characters in the screen data that are unique to the agent. The present invention is a much simpler method of transforming a character screen into a GUI screen that the known art such as Cascio.

Claim 1, for example, as amended to overcome the Section 101 rejection, sets forth the above distinctions over Cascio. For example, the language recites

> *scanning the character based user interface by a <u>plurality of agents</u>; and*
>
> *determining which host component types exist in the character based user interface, <u>each agent determining the existence of a different host component type</u> from the other agents.*
>
> (Underlining added to emphasize the distinctions).

The other independent claims 8 and 15 contain equivalent language.

With all respect, it is further argued that in rejecting the present claims, Examiner has imported a host of features into Cascio that simply are not there. For example, for claim 1, Examiner cites Fig. 7 and paragraphs 25, 47, 55 and 75 for the proposition that Cascio discloses a plurality of agents that each determine a different screen component from the other agents. Fig. 7 discloses prior art to Cascio and is nothing more than an example of a character-bases legacy screen. Paragraph 25 is part of the Summary of Cascio and states the objective of matching rules to screen data. Paragraphs 47, 55 and 75 describe more details of Cascio's implementation, but none of these citations describe a plurality of processes (agents) with each being conditioned to find a single and different screen component in the data. Fig. 6 of Cascio confirms that Cascio uses a single computer process and a table of rules to identify screen components.

There are other claimed distinctions as well. Claim 1 calls for determining overlapping portions of screen components and resolving the conflicts. Examiner alleges that Cascio Fig. 6 and paragraphs 58, 72-73, and 76-77 teach the determination of overlapping regions. However, Cascio contains no such teaching at all. Fig. 6 is a flowchart of Cascio's single process and contains no step of determining overlapping regions. Paragraph 58 describes using both text and field attributes in a rule, but does not address determining overlapping regions; paragraphs 72-73 describe the single process of Fig. 6 and contains nothing relating to overlapping screen components; paragraph 76 points to Fig. 11 as an example of a GUI transformed version of the character screen of Fig. 7. No overlapping components are shown in either Fig. 7 or 11. Paragraph 77 describes the possibility of transforming a legacy screen into different languages, such as XML, HTML and the like. But, again there is no mention of overlapping screen components.

The Section 103 Rejection

Claims 8-14 are rejected a being obvious under Cascio in View of DaCosta. Independent claim 8 contains language similar to the multiple agents (processes) discussed above. A minor variation is that the claim refers to a plurality of object agents instead of just agents. It is known to skilled art workers in the field of programming that applications can be implemented as objects (also known as classes) or, alternatively, as procedural code. A procedural agent would execute as a single process as above discussed. An object can contain one or more "methods", with each method also executing as a single process when called. Therefore, it is argued that claim 8 and it's dependent claims also distinguish over Cascio in the use of multiple objects, each conditioned to match a different data pattern. Applicants agree that the preamble language of claim 8 relating to a system containing an anti-virus program is common knowledge and unnecessary to the claim. Therefore, this language is removed from claim 8.

Dependent claims 2-7 9-14 and 16--21 are allowable because they introduce other distinguishing features that are discussed above, and because of their dependency.

All claims are now believed to distinguish over the art of record and are in condition for allowance. Reconsideration and passage to issue are respectfully requested.

Respectfully Submitted,

/Jeanine Ray-Yartletts/

Jeanine Ray-Yartletts
Reg. No. 39808

Customer Number 25259
IBM Corporation
Department T81/Building 503
P.O. Box 12195
Research Triangle Park, NC 27709

(919) 543-2541
FAX: 919-254-4330
Email: jeanine@us.ibm.com